Yee & Associates, P.C.

Suite 1100
Dallas, Texas 75244

4100 AlphaRoad

Main No. (972) 385-8777
Facsimile (972) 385-7766

Facsimile Cover Sheet

To: Commissioner for Patents for **Examiner Thomas Duong** **Group Art Unit 2143**	Facsimile No.: **703/872-9306**
From: Carrie Parker Legal Assistant to Wayne P. Bailey	No. of Pages Including Cover Sheet: **29**

Message:

Enclosed herewith:

- Transmittal Document; and
- Appeal Brief.

☐

Re: Application No. **09/710,921**
Attorney Docket No: **AUS9-2000-0561-US1**

Date: Wednesday, December 15, 2004

Please contact us at (972) 385-8777 if you do not receive all pages indicated above or experience any difficulty in receiving this facsimile.

PLEASE CONFIRM RECEIPT OF THIS TRANSMISSION BY FAXING A CONFIRMATION TO 972-385-7766.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Hamilton, II et al.** § Group Art Unit: **2143**

Serial No.: **09/710,921** § Examiner: **Duong, Thomas**

Filed: **November 9, 2000** § Attorney Docket No.: **AUS9-2000-0561-US1**

For: **Apparatus and Methods for Sequentially Scheduling a Plurality of Commands in a Processing Environment which Executes Commands Concurrently**

35525
PATENT TRADEMARK OFFICE CUSTOMER NUMBER

Certificate of Transmission Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (703) 872-9306 on December 15, 2004.
By: Carrie Parker
Carrie Parker

TRANSMITTAL DOCUMENT

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:
ENCLOSED HEREWITH:

- Appeal Brief (37 C.F.R. 41.37).

A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

Respectfully submitted,

Cathrine K. Kinslow
Registration No. 51,886
Duke W. Yee
Registration No. 34,285
YEE & ASSOCIATES, P.C.
P.O. Box 802333
Dallas, Texas 75380
(972) 385-8777

RECEIVED
CENTRAL FAX CENTER
DEC 15 2004

Docket No. AUS9-2000-0561-US1 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Hamilton, II et al.** §
§ Group Art Unit: **2143**
Serial No. **09/710,921** §
§ Examiner: **Duong, Thomas**
Filed: **November 9, 2000** §
§
For: **Apparatus and Methods for Sequentially Scheduling a Plurality of Commands in a Processing Environment which Executes Commands Concurrently** §

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Transmission Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (703) 872-9306 on December 15, 2004.

By: Carrie Parker
Carrie Parker

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on October 15, 2004.

The fees required under § 41.20(B)(2), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-33

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: 1, 11 and 21
2. Claims withdrawn from consideration but not canceled: none
3. Claims pending: 2-10, 12-20 and 22-33
4. Claims allowed: none
5. Claims rejected: 2-10, 12-20 and 22-33

C. CLAIMS ON APPEAL

The claims on appeal are: 2-10, 12-20 and 22-33

STATUS OF AMENDMENTS

No amendment after final was filed for this case.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 2 - INDEPENDENT

Claim 2 is directed to a method in a data processing system for scheduling the execution of a plurality of commands, the data processing system including an environment which executes commands concurrently, where the commands are executed without regard to a completion of execution of any other ones of the commands. The method includes the steps of selecting a plurality of commands from the environment which executes commands concurrently, and scheduling execution of the selected plurality of commands in a programming order. The scheduling step includes: (i) encapsulating a first one of the plurality of commands in a first process and encapsulating a second one of the plurality of commands in a second process; (ii) beginning processing of the first process; (iii) executing the first one of the plurality of commands in response to the beginning processing of the first process, wherein the first one of the plurality of commands executes only while the first process is executing; and (iv) beginning processing of the second process only in response to a completion of processing of the first process. This method is described at Specification page 10, line 29 – page 12, line 12 and shown by the process flows depicted in Figures 3 and 4.

B. CLAIM 12 - INDEPENDENT

Claim 12 is directed to a data processing system for scheduling the execution of a plurality of commands, the data processing system including an environment which executes commands concurrently, where the commands are executed without regard to a completion of execution of any other ones of the commands. The system includes a selecting means for selecting a plurality of commands from the environment which executes commands concurrently (the corresponding structure is shown in Figure 2 at 200), and a scheduler for scheduling execution of the selected plurality of commands. The scheduler comprises: (i) means for encapsulating a first one of the plurality of commands in a first process and encapsulating a second one of the plurality of commands in a second process (the corresponding structure is shown in Figure 2 at 200); (ii) means for beginning processing of the first process (the corresponding structure is shown in

Figure 2 at 200); (iii) means for executing the first one of the plurality of commands in response to the beginning processing of the first process, wherein the first one of the plurality of commands executes only while the first process is executing (the corresponding structure is shown in Figure 2 at 200); and (iv) means for beginning processing of the second process only in response to a completion of processing of the first process (the corresponding structure is shown in Figure 2 at 200). This system functionality is described at Specification page 10, line 29 – page 12, line 12 and shown by the process flows depicted in Figures 3 and 4 in conjunction with the hardware shown in Figures 1 (reference number 100) and 2 (reference number 200).

C. CLAIM 22 – INDEPENDENT

Claim 22 is directed to a computer program product for scheduling the execution of a plurality of commands, the data processing system including an environment which executes commands concurrently, where the commands are executed without regard to a completion of execution of any other ones of the commands. The computer program product includes a selecting means for selecting a plurality of commands from the environment which executes commands concurrently, and a scheduler for scheduling execution of the selected plurality of commands. The scheduler comprises: (i) instructions means for encapsulating a first one of the plurality of commands in a first process and encapsulating a second one of the plurality of commands in a second process; (ii) instruction means for beginning processing of the first process; (iii) instruction means for executing the first one of the plurality of commands in response to the beginning processing of the first process, wherein the first one of the plurality of commands executes only while the first process is executing; and (iv) instruction means for beginning processing of the second process only in response to a completion of processing of the first process. This computer program product functionality is described at Specification page 10, line 29 – page 12, line 12 and shown by the process flows depicted in Figures 3 and 4.

D. CLAIM 17 – DEPENDENT MEANS-PLUS-FUNCTION

Claim 17 recites means for assigning a first process identifier to said first process, which is described at Specification page 12, lines 13-26 with reference to Figure 5 and shown by element 100 of Figure 1 and element 200 of Figure 2; and means for utilizing said first process identifier

to determine whether said first process is currently executing, which is described at Specification page 12, lines 13-26 with reference to Figure 5 (blocks 502, 504 and 506) and shown by element 100 of Figure 1 and element 200 of Figure 2.

E. CLAIM 18 – DEPENDENT MEANS-PLUS-FUNCTION

Claim 18 recites means for searching a process table for said first process identifier, which is described at Specification page 12, lines 13-26 with reference to Figure 5 (block 504) and shown by element 100 of Figure 1 and element 200 of Figure 2; means for determining that said first process is executing in response to locating said process identifier in said process table, which is described at Specification page 12, lines 13-26 with reference to Figure 5 (block 506) and shown by element 100 of Figure 1 and element 200 of Figure 2; and means for determining that said first process is not executing in response to a failure to locate said process identifier in said process table, which is described at Specification page 12, lines 27-30 with reference to Figure 5 (block 506) and shown by element 100 of Figure 1 and element 200 of Figure 2.

F. CLAIM 20 – DEPENDENT MEANS-PLUS-FUNCTION

Claim 20 recites means for establishing a timer for said first process, which is described at Specification page 13, lines 3-17 with reference to Figure 5 and shown by element 100 of Figure 1 and element 200 of Figure 2; means for starting said timer in response to executing said first process, which is described at Specification page 13, lines 3-17 with reference to Figure 5 (block 506) and shown by element 100 of Figure 1 and element 200 of Figure 2; and means for testing said return code variable to determine whether said return code variable is equal to said second value upon the expiration of said timer, which is described at Specification page 13, lines 3-17 with reference to Figure 5 (blocks 514, 504 and 506) and shown by element 100 of Figure 1 and element 200 of Figure 2.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. GROUND OF REJECTION 1 (Claims 2-10, 12-20, 22-33)

Claims 2-10, 12-20 and 22-33 stand rejected under 35 U.S.C. § 102(e) as being anticipated by Blelloch et al. (US006434590B1).

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 2-10, 12-20, 22-33)

A.1. Claims 2, 12 and 22 (and dependent claims thereof)

As previously described, the present invention is directed to a technique for selecting a plurality of commands from the environment which executes commands concurrently, and scheduling execution of the selected plurality of commands in a programming order. The cited Blelloch reference teaches exactly the opposite - the ordering of tasks that have been *sequentially* scheduled, and selecting a subset of the available tasks for *parallel* processing. This can be seen by Blelloch's teaching at Col. 4, lines 16-21, where it states:

> "The invention utilizes the ordering of tasks in the **sequential scheduling** to select a subset of the available tasks for **parallel processing**. That is, the invention selects a subset of available tasks for parallel processing by assigning higher priorities to the earlier available tasks in the sequential schedule." (emphasis added by Appellants)

This description is just the opposite of the claimed invention recited in Claim 2, as Claim 2 recites that commands are selected from an environment which executes commands *concurrently* (i.e. in parallel), and these selected commands are then scheduled to execute in a particular *sequential* order. For a prior art reference to anticipate in terms of 35 U.S.C. 102, every element of the claimed invention must be identically shown in a single reference. *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990). As every element of the claimed invention is not identically shown in a single reference – and in particular the cited reference does not teach a method in a data processing system for sequentially scheduling the execution of a plurality of commands that are otherwise executed without regard to a completion of execution of any other ones of said commands – it is shown that Claim 2 is not anticipated by the cited reference.

In addition to the general data processing environment being different – and in fact being just the opposite of what is claimed - the cited Blelloch reference is also deficient in teaching two

particular aspects of the claimed sequential scheduling: (1) a step of encapsulating individual commands into distinct processes, and (2) use of a scheduler to initiate process execution, as will now be described in detail.

Encapsulating Commands

A key enabling feature of the present invention of sequential scheduling of non-ordered commands is the encapsulation of such non-ordered commands into processes to ensure a particular (sequential) execution order. Per Claim 2 (and similarly for Claims 12 and 22), *as a part of the scheduling of command execution*, commands selected from the concurrent execution environment are encapsulated in processes such that a first process completes execution before a second process (that includes an encapsulated second command) begins execution, thus ensuring a particular sequential order of command execution. The Examiner states that since Blelloch teaches that a task is a plurality of commands, each task is an encapsulation of instructions that are to be executed sequentially. However, Appellants urge three-fold error in such position as will now be described.

Per claim 2, *commands are selected from a concurrent execution environment and encapsulated into processes,* in order to provide for sequential execution *of the processes* and their respective encapsulated command ("beginning processing of the said first process", "executing said first one of said plurality of commands in response to said beginning processing of said first process, wherein said first one of said plurality of commands executes only while said first process is executing" and "beginning processing of said second process only in response to a completion of processing of said first process"). Even assuming arguendo that Blelloch's tasks are a plurality of encapsulated commands, these commands are not selected from a concurrent execution environment, as expressly recited in Claim 2. These Blelloch tasks also comprise a plurality of commands with no guaranteed sequential order of execution (due to possible conditional branches, do-loops, input variable dependencies, etc.). Thus, these Blelloch tasks (containing embedded instructions) (1) do not contain commands selected from a concurrent execution environment, and (2) do not guarantee any particular order of execution (i) amongst themselves (they are dispatched to a queue from which a plurality of parallel processors extract tasks therefrom (Blelloch col. 3, lines 46-51) or (ii) of the instructions contained therein.

Further, this alleged Blelloch encapsulation is not done as part of scheduling (the scheduling being done by Blelloch's assignment manager), but rather is done by a preprocessor as part of a compile/translate step (Blelloch Col. 2, lines 28-38) and thus is not a part of scheduling, as claimed. In addition, and as alluded to above, because this alleged Blelloch encapsulation of instructions into a task is a part of the initial compile of an incoming program, there is no "selecting said plurality of commands from the environment which executes commands concurrently", as expressly required by Claim 2 (and similarly for Claims 12 and 22). Rather, a traditional program compile step occurs, which is not a part of task scheduling.

Finally, this interpretation of encapsulating by the Examiner does not result in a teaching of encapsulating a first of the plurality of commands in a first process and encapsulating a second of the plurality of commands in a second process, as per the Examiner's interpretation of Blelloch (each task is an encapsulation of instructions), multiple instructions are encapsulated together into a single task. Claim 2 expressly states encapsulation of individual commands into different processes to thereby ensure proper sequential execution. This proper sequential execution of individually encapsulated commands into unique processes is accomplished by the steps of "beginning processing of the said first process", "executing said first one of said plurality of commands in response to said beginning processing of said first process, wherein said first one of said plurality of commands executes only while said first process is executing" and "beginning processing of said second process only in response to a completion of processing of said first process".

As to Blelloch's task queue TQ1 (Fig 4), and even assuming that tasks which are placed on such queue may be sequentially executed, these tasks have not been encapsulated into a process after having been selected from an environment of tasks that otherwise execute concurrently. In other words, they are executed in the same task form as selected, without subsequent encapsulation, as claimed. Thus, even this teaching does not satisfy the teaching deficiency regarding the specifics of the claimed encapsulation processing.

As shown above, the claimed encapsulation technique is substantially different from a mere task having a plurality of instructions. Thus, it has been shown that the cited reference does not teach the claimed command encapsulation technique. Appellants will now address the second key enabling feature for providing sequential scheduling of non-ordered commands.

Use of a Scheduler to Initiate Process Execution

Another key enabling feature of the present invention of sequential scheduling of non-ordered commands is the use of a scheduler to initiate process execution. Per Claim 2 (and similarly for Claims 12 and 22), *as a part of the scheduling of command execution*, a first process having a command encapsulated therein is started, and a second process having a command encapsulated therein is then started in response to completion of processing of the first process. This ensures a particular order of command execution for the command encapsulated within each process. The Blelloch reference teaches a system having two subsets of processors – 'worker' processors which execute tasks and 'scheduler' processors which execute the scheduler (Col. 14, lines 4-42). All of the processing elements operate separately without being in synchronization (Col 5, lines 19-21). While it appears that the scheduler initially determines a sequential ordering of tasks for processing (Abstract 1-4), this sequential schedule is independent of the actual parallel execution that occurs (Abstract lines 10-15). The scheduled tasks are placed on a queue, where they are extracted for execution by the 'worker' processors (Col. 2, lines 43-51), thereby creating a pull system rather than a push system (Col. 8, lines 17-21). This independence between the scheduler and process execution is expressly desired to allow tasks to run in parallel and independent of one another (Col. 1, lines 36-48), and to improve overall system flexibility (Col. 5, lines 19-21). Thus, it is shown that the cited reference does not teach a scheduler or scheduling step that includes process invocation.

Further regarding the details of this claimed scheduling step, the cited reference does not teach "beginning processing of said second process only in response to a completion of processing of said first process". While Blelloch teaches that there are synchronization variables that can cause a given process thread to wait on another thread to set a synchronization variable (Col. 8, lines 43-59), these 'waiting' threads have already begun execution and are subsequently placed in a wait state, and are awakened when another process thread writes to the synchronization variable (Col. 10, lines 44-65). Thus, these synchronization variables do not teach the claimed step of "beginning processing of said second process only in response to a completion of processing of said first process" as both the spawned thread and waiting thread have begun processing and are thus executing concurrently.

Thus, Claim 2 (and similarly for Claims 12 and 22) is further shown to have been erroneously rejected under 35 U.S.C. 102, and every element of the claimed invention is not identically shown in a single reference.

A.2. Claims 7, 17 and 27

Further with respect to dependent Claims 7, 17 and 27 (which due to their dependencies on independent Claims 2, 12 and 22, respectively, are not anticipated for reasons given above with respect to such independent claims), Appellants urge that the cited reference does not teach the claimed steps of assigning a first process identifier to said first process, and utilizing said first process identifier to determine whether said first process is currently executing. In rejecting these claims, the Examiner cites Blelloch abstract; col. 1, lines 5-9; col. 9, lines 14-40; col. 10, lines 35-64; col. 13, lines 46-65; col. 14, lines 21-42; fig. 5-8. Appellants have thoroughly reviewed all such cited passages, and can find no mention of use of a process identifier *to determine if a process is currently executing*. Thus, it is further shown that Claims 7, 17 and 27 have been erroneously rejected under 35 U.S.C. 102, as every element of the claimed invention is not identically shown in a single reference.

A.3. Claims 8, 18 and 28

Further with respect to dependent Claims 8, 18 and 28 (which due to their dependencies on independent Claims 2, 12 and 22, respectively, are not anticipated for reasons given above with respect to such independent claims; and due to their dependencies on dependent Claims 7, 17 and 27, respectively, are not anticipated for further reasons given above with respect to such dependent claims), Appellants urge that the cited reference does not teach the claimed steps of searching a process table for said first process identifier, determining that said first process is executing in response to locating said process identifier in said process table, and determining that said first process is not executing in response to a failure to locate said process identifier in said process table. In rejecting these claims, the Examiner makes no assertion as to any use of a process table by the teachings of the cited Blelloch reference. Thus, the Examiner has failed to establish any teaching by Blelloch of use of a process table to determine if a process is

executing[1]. Thus, it is further shown that Claims 8, 18 and 28 have been erroneously rejected under 35 U.S.C. 102, as every element of the claimed invention is not identically shown in a single reference.

A.4. Claims 10, 20 and 30

Further with respect to dependent Claims 10, 20 and 30 (which due to their dependencies on independent Claims 2, 12 and 22, respectively, are not anticipated for reasons given above with respect to such independent claims; and due to their ultimate dependencies on dependent Claims 7, 17 and 27, respectively, are not anticipated for further reasons given above with respect to such dependent claims; and due to their ultimate dependencies on dependent Claims 8, 18 and 28, respectively, are not anticipated for further reasons given above with respect to such dependent claims), Appellants urge that the cited reference does not teach the claimed steps of establishing a timer for said first process, starting said timer in response to executing said first process, and testing said return code variable to determine whether said return code variable is equal to said second value upon the expiration of said timer. In rejecting these claims, the Examiner makes no assertion as to any use of a timer by the teachings of Blelloch. Thus, the Examiner has failed to establish any teaching by Blelloch of use of a timer to assist in determining if a return code variable is equal to a particular (second) value as a part of determining whether the first process is executing. Thus, it is further shown that Claims 10, 20 and 30 have been erroneously rejected under 35 U.S.C. 102, as every element of the claimed invention is not identically shown in a single reference.

A.5. Claims 31, 32 and 33

Further with respect to dependent Claims 31, 32 and 33 (which due to their dependencies on independent Claims 2[2], 12 and 22, respectively, are not anticipated for reasons given above with respect to such independent claims), Appellants urge that the cited reference does not teach the claimed feature of wherein said first process and second process are included in a script. In rejecting these claims, the Examiner makes no assertion as to any inclusion of processes in a

[1] In making such final rejection, the examiner shall repeat or state all grounds of rejection then considered applicable to the claims in the application, clearly stating the reasons in support thereof. MPEP 706.07

[2] There is an obvious typographical error regarding Claim 31, which should depend upon Claim 2 as Claim 1 has previously been cancelled.

script by the teachings of Blelloch. Thus, the Examiner has failed to establish any teaching by Blelloch of use of a script, or inclusion of first and second processes in such (missing) script. Thus, it is further shown that Claims 31, 32 and 33 have been erroneously rejected under 35 U.S.C. 102, as every element of the claimed invention is not identically shown in a single reference.

CONCLUSION

It has thus been shown that there are numerous claimed features not taught by the cited reference, and thus the Examiner has erroneously rejected all claims under 35 USC 102(e). Accordingly, Appellants urge that the Board reverse the Examiner's rejection of Claims 2-10, 12-20 and 22-33.

Cathrine K. Kinslow
Reg. No. 51,886
Wayne P. Bailey
Reg. No. 34,289
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

2. A method in a data processing system for scheduling the execution of a plurality of commands, said data processing system including an environment which executes commands concurrently, wherein said commands are executed without regard to a completion of execution of any other ones of said commands, said method comprising the steps of:

selecting said plurality of commands from the environment which executes commands concurrently;

scheduling execution of said selected plurality of commands in a programming order, said scheduling step comprising:

encapsulating said first one of said plurality of commands in a first process and encapsulating said second one of said plurality of commands in a second process;

beginning processing of said first process;

executing said first one of said plurality of commands in response to said beginning processing of said first process, wherein said first one of said plurality of commands executes only while said first process is executing; and

beginning processing of said second process only in response to a completion of processing of said first process.

3. The method according to claim 2, further comprising the step of completing processing of said first process in response to a completion of execution of said first one of said plurality of commands.

4. The method according to claim 2, further comprising the step of executing said second one of said plurality of commands in response to said beginning processing of said second process.

5. The method according to claim 2, further comprising the step of determining whether said first process is currently executing.

6. The method according to claim 5, wherein said step of determining whether said first process is currently executing further comprises the steps of:

establishing a return code variable; and

utilizing said return code variable to indicate whether said first process is currently executing.

7. The method according to claim 6, wherein said step of determining whether said first process is currently executing further comprises the steps of:

assigning a first process identifier to said first process; and

utilizing said first process identifier to determine whether said first process is currently executing.

8. The method according to claim 7, further comprising the steps of:

searching a process table for said first process identifier;

determining that said first process is executing in response to locating said process identifier in said process table; and

determining that said first process is not executing in response to a failure to locate said process identifier in said process table.

9. The method according to claim 8, further comprising the steps of:

setting said return code variable equal to a first value while said first process is executing; and

setting said return code variable equal to a second value when said first process has completed executing.

10. The method according to claim 9, further comprising the steps of:

establishing a timer for said first process;

starting said timer in response to executing said first process; and

testing said return code variable to determine whether said return code variable is equal to said second value upon the expiration of said timer.

12. A data processing system for scheduling the execution of a plurality of commands, said data processing system including an environment which executes commands concurrently, wherein said commands are executed without regard to a completion of execution of any other ones of said commands, comprising:

selecting means for selecting said plurality of commands from the environment which executes commands concurrently;

a scheduler for scheduling execution of said selected plurality of commands in said environment, said scheduler comprising:

means for encapsulating said first one of said plurality of commands in a first process and encapsulating said second one of said plurality of commands in a second process;

means for beginning processing of said first process;

means for executing said first one of said plurality of commands in response to said beginning processing of said first process, wherein said first one of said plurality of commands executes only while said first process is executing; and

means for beginning processing of said second process only in response to a completion of processing of said first process.

13. The system according to claim 12, further comprising means for completing processing of said first process in response to a completion of execution of said first one of said plurality of commands.

14. The system according to claim 12, further comprising means for executing said second one of said plurality of commands in response to said beginning processing of said second process.

15. The system according to claim 12, further comprising means for determining whether said first process is currently executing.

16. The system according to claim 15, wherein said means for determining whether said first process is currently executing further comprises:

means for establishing a return code variable; and

means for utilizing said return code variable to indicate whether said first process is currently executing.

17. The system according to claim 16, wherein said means for determining whether said first process is currently executing further comprises:

means for assigning a first process identifier to said first process; and

means for utilizing said first process identifier to determine whether said first process is currently executing.

18. The system according to claim 17, further comprising:

means for searching a process table for said first process identifier;

means for determining that said first process is executing in response to locating said process identifier in said process table; and

means for determining that said first process is not executing in response to a failure to locate said process identifier in said process table.

19. The system according to claim 18, further comprising:

means for setting said return code variable equal to a first value while said first process is executing; and

means for setting said return code variable equal to a second value when said first process has completed executing.

20. The system according to claim 19, further comprising:

means for establishing a timer for said first process;

means for starting said timer in response to executing said first process; and

means for testing said return code variable to determine whether said return code variable is equal to said second value upon the expiration of said timer.

22. A computer program product for scheduling the execution of a plurality of commands, said data processing system including an environment which executes commands concurrently, wherein said commands are executed without regard to a completion of execution of any other ones of said commands, said computer program product comprising:

selecting means for selecting said plurality of commands from the environment which executes commands concurrently;

a scheduler for scheduling execution of said selected plurality of commands in said environment, said scheduler comprising:

instruction means for encapsulating said first one of said plurality of commands in a first process and encapsulating said second one of said plurality of commands in a second process;

instruction means for beginning processing of said first process;

instruction means for executing said first one of said plurality of commands in response to said beginning processing of said first process, wherein said first one of said plurality of commands executes only while said first process is executing; and

instruction means for beginning processing of said second process only in response to a completion of processing of said first process.

23. The computer program product according to claim 22, further comprising instruction means for completing processing of said first process in response to a completion of execution of said first one of said plurality of commands.

24. The computer program product according to claim 22, further comprising instruction means for executing said second one of said plurality of commands in response to said beginning processing of said second process.

25. The computer program product according to claim 22, further comprising instruction means for determining whether said first process is currently executing.

26. The computer program product according to claim 25, wherein said instruction means for determining whether said first process is currently executing further comprises:

instruction means for establishing a return code variable; and

instruction means for utilizing said return code variable to indicate whether said first process is currently executing.

27. The computer program product according to claim 26, wherein said instruction means for determining whether said first process is currently executing further comprises:

instruction means for assigning a first process identifier to said first process; and

instruction means for utilizing said first process identifier to determine whether said first process is currently executing.

28. The computer program product according to claim 27, further comprising:

instruction means for searching a process table for said first process identifier;

instruction means for determining that said first process is executing in response to locating said process identifier in said process table; and

instruction means for determining that said first process is not executing in response to a failure to locate said process identifier in said process table.

29. The computer program product according to claim 28, further comprising:

instruction means for setting said return code variable equal to a first value while said first process is executing; and

instruction means for setting said return code variable equal to a second value when said first process has completed executing.

30. The computer program product according to claim 29, further comprising:

instruction means for establishing a timer for said first process;

instruction means for starting said timer in response to executing said first process; and

instruction means for testing said return code variable to determine whether said return code variable is equal to said second value upon the expiration of said timer.

31. The method according to Claim 1, wherein said first process and said second process are included in a script.

32. The system according to Claim 12, wherein said first process and said second process are included in a script.

33. The computer program product according to Claim 22, wherein said first process and said second process are included in a script.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.